CSC 111:
Intro to Computer Science through Programming

Spring 2017
Prof. Sara Mathieson
Admin

- No Lab 4 due to Rally Day
- Quiz solutions are posted
- **Start Homework 3 early!**
- You should be spending roughly 10 hours per week on CSC 111 outside of class/lab
Lab reminders

- Make sure to come to lab and come on time
  - Lab attendance is required and more than one absence will affect your lab grade (unless you have a deans notice)

- Work with your partner
  - It is unfair to your partner if they never get the chance to code

- Talk to your partner
  - Part of the goal of lab is not only to program, but to learn to communicate using this new language
Outline: 2/17

- Feedback from Monday
- Continue guess my number program
- Recap string/list parallels
- Slicing strings and lists
Feedback from Monday
Monday anonymous questions

- Pace seems mostly okay, but a bit fast during live coding (too fast switching between code and slides)
- Forming study groups (“Search for Teammates” on Piazza)
- Coding techniques and how to approach problems
- More ungraded practice problems (try the book exercises)
- TA hours are busy (try Wed/Thurs, try to find others working on the same part of the homework)
- Go over homework and labs
Homework 3 Preview: ELIZA

http://www.masswerk.at/elizabot/
Guess my number
(live coding)
Guess my number extensions

+ Use a Boolean variable to “flip” the state from not having guessed the answer to having guessed the answer

+ Compute the number of tries it took the computer to guess

Enter a integer between 0 and 9 inclusive: 6
8 is wrong, guess again.
1 is wrong, guess again.
8 is wrong, guess again.
0 is wrong, guess again.
You guessed my number!
You took 4 tries to guess my number.

Enter a integer between 0 and 9 inclusive: 7
5 is wrong, guess again.
4 is wrong, guess again.
3 is wrong, guess again.
8 is wrong, guess again.
6 is wrong, guess again.
8 is wrong, guess again.
9 is wrong, guess again.
0 is wrong, guess again.
2 is wrong, guess again.
1 is wrong, guess again.
3 is wrong, guess again.
1 is wrong, guess again.
3 is wrong, guess again.
6 is wrong, guess again.
You guessed my number!
You took 14 tries to guess my number.

Enter a integer between 0 and 9 inclusive: 8
9 is wrong, guess again.
6 is wrong, guess again.
5 is wrong, guess again.
3 is wrong, guess again.
9 is wrong, guess again.
You guessed my number!
You took 5 tries to guess my number.
Recap parallels between strings and lists
## Similarities between list and str

<table>
<thead>
<tr>
<th></th>
<th>Assign a value</th>
<th>Concatenate + sign</th>
<th>Indexing [square brackets]</th>
<th>Get type</th>
</tr>
</thead>
<tbody>
<tr>
<td>lst = [&quot;zero&quot;,&quot;one&quot;,&quot;two&quot;,&quot;three&quot;,&quot;four&quot;,&quot;five&quot;]</td>
<td>string = &quot;Smith College&quot;</td>
<td>lst + lst2 = [&quot;six&quot;,&quot;seven&quot;,&quot;eight&quot;]</td>
<td>lst[3] = 'three'</td>
<td>&gt;&gt;&gt; type(lst) &lt;class 'list'&gt;</td>
</tr>
<tr>
<td></td>
<td>length</td>
<td>string2 = &quot; is in Northampton&quot;</td>
<td>lst[0] = 'zero'</td>
<td>&gt;&gt;&gt; lst.count(&quot;two&quot;) 1</td>
</tr>
<tr>
<td>len(lst) = 6</td>
<td>len(string) = 13</td>
<td>string + string2 = 'Smith College is in Northampton'</td>
<td>string[3] = 't'</td>
<td>&gt;&gt;&gt; type(string) &lt;class 'str'&gt;</td>
</tr>
<tr>
<td>&gt;&gt;&gt; type(lst) &lt;class 'list'&gt;</td>
<td>&gt;&gt;&gt; string.count(&quot;e&quot;) 2</td>
<td>&gt;&gt;&gt; string[0] = 's'</td>
<td>&gt;&gt;&gt; lst.count(&quot;two&quot;) 1</td>
<td></td>
</tr>
</tbody>
</table>
Strings are special though...

```python
>>> lst.split()
Traceback (most recent call last):
  File "<pyshell#39>", line 1, in <module>
    lst.split()
AttributeError: 'list' object has no attribute 'split'

>>> string.split()
["Smith", "College"]

>>> string.lower()
'smith college'

>>> string.upper()
'SMITH COLLEGE'

>>> string.replace(" ","")
'SmithCollege'
```
“Slicing and dicing”
Idea: create a substring or sublist

Still use square brackets, but with start and stop elements separated by a colon:

```python
>>> college = "Smith College"
>>> college[0]
'S'
>>> college[4]
'h'
>>> college[0:4]
'Smit'
>>> # slicing is like range: include start but exclude end
```

If start is omitted, start from the beginning (zero\textsuperscript{th} element), and if end is omitted, go until the very end

```python
>>> college[:4]
'Smit'
>>> college[5:8]
'Co'
>>> college[6:len(college)]
'College'
>>> college[6:]
'College'
```
Exercise: months of the year

+ Write a function to print the month abbreviation, given the month number

```
months = "JanFebMarAprMayJunJulAugSepOctNovDec"

>>> month_name(4)
The number 4 month is Apr
>>> month_name(1)
The number 1 month is Jan
>>> month_name(9)
The number 9 month is Sep
>>> month_name(2)
The number 2 month is Feb
```

# given a month number n (1,2,...,12), print the
# 3 letter month abbreviation

```python
def month_name(n):
    months = "JanFebMarAprMayJunJulAugSepOctNovDec"
    return months[n-1:n+2]
```

```python
>>> month_name(4)
"Apr"
>>> month_name(1)
"Jan"
>>> month_name(9)
"Sep"
>>> month_name(2)
"Feb"
```
Why start numbering from zero?

Argument from influential computer scientist Dijkstra:

http://www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html